**CSCI 1301 Lecture 5**

Key points of this lecture:  
1- Arithmetic Operations  
2- Operator Precedence  
3- Increment and Decrement Operators  
4- Interactive programming  
Expressions:  
  
an expression is a combination of one or more operators and operands. Arithmetic expressions compute numeric results and make use of the arithmetic operations. Arithmetic operations are addition "+", subtraction "-", multiplication "\*", division "/" and remainder "%". If either or both operands are floating-point values, the result is floating-point value.  
  
**Arithmetic Operations**:  
  
Addition, subtraction, and multiplication are typical mathematical processes or Arithmetic Operations.   
int x = 10, y = 20, z;  
  
z = x + y; *the value of z is 30 (z = 10 + 20)  
z = y – x;* the value of z is 10 (z = 20 – 10)  
z = x \* y; // the value of z is 200 (z = 10 \* 20)  
  
Division and remainder are a little different. If both operands are integers, the division result will be an integer and result will ignore the fraction part. For example 4/3 = 1 and 7/9 = 0  
If one or both operands are floats or doubles, the result will be a typical mathematical division.  
  
The remainder “%”will print the fraction part of the division operation. For example 14%3 = 2 and 15%5 = 0 and 8%12 = 8  
  
**Operator Precedence**:  
  
Operator Precedence is the rule that establishes the order of operation. Multiplication, division and remainder are performed before addition and subtraction. Arithmetic operators with the same precedence are evaluated from left to right. The left most one is performed first.  
Parentheses are the highest precedence. If an operator is within parentheses it will be performed first regardless what it is.  
  
1 + 2 + 3 \* 4 = 1 + 2 + 12 = 15  
3 \* 3 + 1 = 9 + 1 = 10  
3 \* 4 / 6 + 4 = 12 / 6 + 4 = 2 + 4 = 6  
  
3 \* 4 / (6 + 4) = 12 / 10 = 1.2  
  
  
5 + 10 \* 2 – 8 / 4 = 5 + 20 – 2 = 23  
(3 + 9) \* (7 – 5) / 4 = 12 \* 2 / 4 = 24 / 4 = 6  
  
**Increment and decrement**:  
  
int count = 10, total;  
count = count + 1; is equivalent to count ++;  
count = count -1; is equivalent to count --;  
total += count; is equivalent to total = total + count;  
total -= count; is equivalent to total = total – count;  
  
**Interactive programming**:  
  
Scanner class is an easy way to receive various types of inputs to the program. System.in represents input from the keyboard. The following is a statement that creates an object that can access class Scanner and read entries from the Keyboard:  
  
Scanner userInput = new Scanner (System.in);  
The new object created, userInput, can access the Scanner class and use, invoke, its methods. Input methods from Scanner class such as nextLine(), nextInt(), nextDouble(). As indicated by the type, nextInt(); will assign the type int to the entry, nextDouble(); will receive a double type, nextChar(); will receive a character type, and nextLine(); will receive a String.  
  
The Scanner class is part of java.Util class library and must be imported to the program for the program to be able to access and use Scanner class. Consider the following sample program to illustrate interactive programming and the use of Scanner class:  
  
**import java.util.Scanner;**  
  
**public class MyTime**  
  
**{**  
  
**public static void main (String[] args)**  
  
**{**  
  
**int distance;**  
  
**double speed, time;**  
**Scanner enteredData = new Scanner (System.in);**  
  
**System.out.print ("Enter the distance in miles: ");**  
  
**distance = enteredData.nextInt();**  
  
**System.out.print ("Enter the speed: ");**  
  
**speed = enteredData.nextDouble();**  
  
**time = distance / speed;**  
  
**System.out.println ("The time required to reach destination is: " +** **time);**  
  
**}**  
  
**}**